Dynamic Uncertainty Estimation for Offline Reinforcement Learning

Jiesheng Wang[1], Lin Li[1], Wei Wei[1*], Yujia Zhang[1], Xin Yang[2]

[1] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi, China

[2] School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, Sichuan, China

Abstract

Offline reinforcement learning confronts the distributional shift challenge, a consequence of learning policy from static datasets. Current methods primarily handle this issue by aligning the learned policy with the behavior policy or conservatively estimating Q-values for out-of-distribution (OOD) actions. However, these approaches can lead to overly pessimistic estimation of Q-values of the OOD actions in unfamiliar situations, resulting in a suboptimal policy. To address this, we propose a new method, Dynamic Uncertainty estimation for Offline Reinforcement Learning. This method introduces a base density-truncated OOD data sampling approach to reduce the impact of extrapolation errors on uncertainty estimation. It enables conservative estimation of Q-values for OOD actions while avoiding negative impacts on in-distribution data. We also develop a dynamic uncertainty estimation mechanism to prevent excessive pessimism and enhance the generalization of the Q-function. This mechanism dynamically adjusts the degree of pessimism in the Q-function by minimizing the error between target and estimated values. Our method outperforms existing algorithms, as demonstrated by experimental results based on the D4RL benchmark, and proves its superiority in addressing the distributional shift challenge.

Introduction

In recent years, deep reinforcement learning (DRL) has achieved significant success in various domains, such as gaming (Silver et al. 2017) and industrial automation (Chen et al. 2019). DRL trains agents through extensive trial-and-error interactions with the environment. However, such trial-and-error interactions are costly and dangerous in fields like healthcare (Yu et al. 2021) and autonomous driving (Hoel, Wolff, and Laine 2023). Offline reinforcement learning (RL) offers a promising direction for the practical application of RL by learning optimal policies from pre-collected data, thus avoiding direct interactions between the agent and the environment.

Due to the extrapolation errors caused by distributional shift (Fujimoto, Meger, and Precup 2019), online RL methods cannot be directly applied to offline RL. Policy constraint and value constraint are employed for model-free offline RL to address this issue. Policy constraint methods, such as BEAR (Kumar et al. 2019), TD3-BC (Fujimoto and Gu 2021), UWAC (Wu et al. 2021), and EMAQ (Ghasemipour, Schuurmans, and Gu 2021) alleviate distributional shift by making the learned policy as close as possible to the behavior policy.

Policy constraint methods limit the optimization space of the learned policy, making it difficult to surpass the behavior policy. Conversely, value constraint methods typically achieve conservative estimation of the Q-function, alleviating extrapolation errors caused by distributional shifts and enhancing the learned policy. For instance, BRAC (Wu, Tucker, and Nachum 2019) employs a behavior regularization term to estimate the Q-function conservatively. CQL (Kumar et al. 2020) directly minimizes the Q-values of out-of-distribution (OOD) actions. IQL (Kostrikov, Nair, and Levine 2021) conservatively estimates Q-function by avoiding querying OOD actions. ATAC (Cheng et al. 2022) uses adversarial training to constrain OOD actions. CSVE (Chen et al. 2023) regularizes OOD actions by conservatively estimating the state value function. RND (Nikulin et al. 2023) and DRND (Yang et al. 2024) mitigate extrapolation errors through the use of random networks. However, these methods can lead to overly conservative Q-functions, which affect the improvement of the policy.

In recent years, methods that estimate uncertainty through an ensemble of Q-networks to conservatively estimate the Q-values of OOD actions have been gaining traction. EDAC (An et al. 2021) estimates uncertainty by using a large ensemble of Q-networks. PBRL (Bai et al. 2022) applies uncertainty penalty separately to OOD actions and in-distribution data by sampling additional OOD data. While these methods mitigate extrapolation errors to some extent, they lack precise characterization of OOD data, which can negatively affect in-distribution data and make it difficult to provide accurate uncertainty estimation. Moreover, in their pursuit of effective OOD actions regularization, existing methods often overlook the importance of generalization for policy improvement, which results in excessively pessimistic penalty that severely impact the generalization of the Q-function, hindering policy enhancement, ultimately leading to the agent frequently ending up learning suboptimal or even inferior policy. Therefore, dynamically adjusting the degree of conservatism of the Q-function during

*Corresponding author: weiwei@sxu.edu.cn

training is crucial for effectively regularizing OOD actions while avoiding excessive pessimism and enhancing the generalization of the Q-function.

However, training Q-functions using only pre-collected offline datasets cannot effectively regularize OOD actions, making it difficult to obtain accurate uncertainty estimation. To address this issue, we propose incorporating OOD data into the training buffer through OOD data sampling. However, existing OOD data sampling methods lack precise OOD data characterization and struggle to distinguish it, potentially harming the Q-function values of in-distribution data. To effectively regularize OOD actions and enhance the reliability of uncertainty estimation, we propose a base density-truncated OOD data sampling approach (DT). By utilizing a reliable explicit density estimation method provided by Flow-GAN (Grover, Dhar, and Ermon 2018), DT can effectively distinguish OOD actions, regularize OOD actions to alleviate extrapolation errors, and make uncertainty estimation more accurate. In addition, we propose a dynamic uncertainty estimation mechanism designed to avoid excessive pessimism and enhance the generalization of the Q-function by constructing auxiliary target values and implementing dynamic uncertainty penalty, which is inspired by recent advancements in online dynamic value function estimation GPL (Cetin and Celiktutan 2023) and CEILING (Zhang et al. 2024a). To this end, by integrating the dynamic uncertainty estimation mechanism with DT, we propose Dynamic Uncertainty Estimation for Offline RL (DURL).

Our work makes three main contributions:

- Proposing a base density-truncated OOD data sampling approach to limit extrapolation errors and improve uncertainty estimation accuracy effectively.
- Proposing a dynamic uncertainty estimation method to avoid excessive pessimism and enhance the generalization of the Q-function in model-free offline RL.
- Extensive experiments were conducted on the D4RL benchmark test, demonstrating the superior performance of DURL compared to existing algorithms.

Related Work

In model-free offline RL, there are mainly two types of methods: policy constraint and value constraint. Policy constraint methods aim to make the learned policy closer to the behavior policy, thereby avoiding performance degradation caused by agent-taking OOD actions. For example, BCQ (Fujimoto, Meger, and Precup 2019) models the behavior policy using generative models to make the learned policy closer to the behavior policy. BEAR (Kumar et al. 2019) measures the difference between the behavior policy and the learned policy using the Maximum Mean Discrepancy distance, ensuring that the difference does not exceed a certain threshold. UWAC (Wu et al. 2021) uses Monte Carlo Dropout for uncertainty estimation and employs an uncertainty weighting mechanism to reduce the impact of distributional shift on policy improvement. TD3-BC (Fujimoto and Gu 2021) adds a behavior cloning regularization term to the loss function of the value function, making the learned policy closer to the behavior policy. However, policy constraint methods overly rely on the behavior policy, limiting the space for policy improvement and making it difficult for the learned policy to surpass the behavior policy.

In value constraint methods, CQL (Kumar et al. 2020) and PSPI (Xie et al. 2021) address overestimation errors by adding a conservative regularization term to estimate the Q-values of OOD actions conservatively. SAC-N (An et al. 2021) estimates uncertainty using an ensemble of numerous Q-networks. EDAC (An et al. 2021) builds upon SAC-N by increasing the diversity within the Q-network ensemble to reduce the number of required Q-networks. However, some datasets still necessitate a large number of Q-network ensembles. PBRL (Bai et al. 2022) employs an ensemble of Q-networks to estimate uncertainty and applies additional OOD data sampling to conservatively estimate the Q-values for OOD actions and in-distribution actions separately, requiring fewer networks compared to EDAC. MCQ (Lyu et al. 2022) regularizes OOD actions by assigning mild but sufficiently conservative pseudo target values to each OOD action. CSVE (Chen et al. 2023) regularizes OOD actions by conservatively estimating the state value function. However, these methods often struggle to effectively distinguish OOD actions and tend to result in excessively conservative Q-function estimation, which impairs the generalization of the Q-function and hinders policy improvement. In contrast, our approach leverages DT and a dynamic uncertainty estimation mechanism to effectively regularize OOD actions while avoiding excessive pessimism, thereby maintaining better generalization of the Q-function.

In model-based offline RL, uncertainty estimation methods are also widely utilized. MOPO (Yu et al. 2020) learns multiple environment models and estimates uncertainty by evaluating the prediction discrepancies among these models, thereby applying penalty to the reward function. MOReL (Kidambi et al. 2020) introduces an uncertainty boundary to ensure the safety and robustness of the policy. TATU (Zhang et al. 2023) leverages uncertainty estimation to penalize the reward function and employs it for data augmentation. OCEAN (Wu et al. 2024) performs conservative exploration through uncertainty estimation. However, these methods share similar limitations with model-free approaches such as SAC-N, EDAC, and PBRL.

In recent years, the application of generative models in offline reinforcement learning has garnered increasing attention. BCQ (Fujimoto, Meger, and Precup 2019) utilizes a Variational Autoencoder to estimate the behavior policy density, aiming to align the learned policy closely with the behavior policy. SPOT (Wu et al. 2022) employs a Conditional Variational Autoencoder to learn the behavior policy density, guiding the policy improvement process. SFBC (Chen et al. 2022) and Diffuser (Janner et al. 2022) use diffusion models to estimate the behavior policy density. However, these methods implicitly estimate the behavior policy density and cannot provide precise density estimation. Flow-GAN (Grover, Dhar, and Ermon 2018), a variant of GAN (Goodfellow et al. 2014), integrates Maximum Likelihood Estimation (MLE) with adversarial learning to deliver more accurate density estimation.

Preliminaries

RL is a machine learning paradigm where an agent learns to maximize cumulative rewards through interactions with its environment. This process can be modeled as a Markov Decision Process (MDP). An MDP is a mathematical model used to describe RL environments, represented by a 6-tuple: $\langle S, A, R, P, \rho, \gamma \rangle$, where S denotes the state space, A denotes the action space, R represents the reward function, $P(s'|s,a)$ is the state transition function, ρ denotes the initial state distribution, and γ is the discount factor ranging from [0,1]. The goal of RL is to learn an optimal policy $\pi^*(a|s)$ that maximizes the expected cumulative reward $J = \mathbb{E}\left[\sum \gamma^t r_t\right]$. The state-action value function $Q_\pi(s,a)$ represents the discounted return obtained by following the current policy π after taking action a in state s. Actor-Critic is a widely used algorithmic framework in RL, comprising two main components: policy evaluation and policy improvement. The objective of the policy evaluation component is to minimize the following loss function:

$$L(\theta) = \mathbb{E}\left[Q_\theta(s,a) - TQ_{\theta^-}(s',a')\right],$$

where T is a Bellman operator defined as $T = r(s,a) + \gamma\mathbb{E}\left[Q_{\theta^-}(s',a')\right]$, θ are the parameters of the Q-function, and θ^- are the parameters of the target Q-network (Osband et al. 2016).

The policy improvement component aims to update the policy in a direction that maximizes the Q-function:

$$L(\psi) = \max \mathbb{E}\left[Q_\theta(s,a)\right],$$

where ψ are the parameters of the policy π.

Offline RL is a subfield of RL where the agent does not directly interact with the environment but instead learns from a pre-collected dataset $D = \{(s_t, a_t, r_t, s_{t+1}, d_t)\}_i$. Here, d_t denotes the termination signal, and actions a_t are generated by a behavior policy μ. Offline RL aims to learn an optimal policy from the offline dataset D. Applying standard RL algorithms such as SAC (Haarnoja et al. 2018), QMD3 (Wei et al. 2022), and RD3 (Zhang et al. 2024b) to offline RL can be challenging due to the issue of extrapolation error.

Method

In this section, we first introduce explicit methods for estimating the behavior policy density. We then describe how to regularize OOD actions and avoid overly pessimistic Q-function estimation through dynamic uncertainty estimation mechanisms and DT. Finally, we provide a detailed implementation of our algorithm.

Explicit Density Estimation

GAN (Goodfellow et al. 2014) have achieved significant success in various domains, including image generation (Radford, Metz, and Chintala 2015), style transfer (Zhu et al. 2017), and data augmentation (Antoniou, Storkey, and Edwards 2017). A GAN comprises two primary components: a generator (G) and a discriminator (D). These components are trained adversarially to produce high-quality data. The optimization objective of a GAN is formulated as follows:

$$\max_\nu \mathbb{E}_{x\sim p_{data}}[\log D_\nu(x)] + \mathbb{E}_{x\sim p_\phi}[\log(1 - D_\nu(x))], \quad (1)$$

where p_{data} denotes the distribution of real data, p_ϕ represents the distribution of generated samples, ϕ refers to the parameters of the generator G, and ν denotes the parameters of the discriminator D. The application of GAN in offline RL is gaining increasing attention (Yang et al. 2022b) (Yang et al. 2022a). However, traditional GAN do not provide explicit density estimation methods, making it challenging to estimate the density of behavior policy accurately. Normalizing Flows (Dinh, Krueger, and Bengio 2014) represent a class of generative models that map simple prior distributions (such as Gaussian distributions) to complex target distributions via invertible transformations, and provide a direct method for calculating log-likelihood, which is essential for density estimation. However, Normalization Flows face challenges when dealing with high-dimensional data.

Flow-GAN integrates Normalizing Flows as the generator within a GAN framework, leveraging the strengths of both generative adversarial networks and Normalizing Flows. This approach not only enables high-quality sample generation but also facilitates effective density estimation.

Assume that the Flow-GAN generator G_ϕ is an invertible transformation in $\mathbb{R}^d$, implying the existence of an inverse function $h_\phi = G_\phi^{-1}$. Let τ denote a trajectory in the offline dataset D. The probability density of τ on the offline dataset D can be expressed as follows:

$$\zeta_\phi(\tau) = \zeta(h_\phi(\tau)) \left| \det \frac{\partial h_\phi(\tau)}{\partial \phi} \right|, \quad (2)$$

ζ represents the probability density of the prior distribution, while ζ_ϕ denotes the probability density of the target distribution. The term $\frac{\partial h_\phi(\tau)}{\partial \phi}$ represents the Jacobian matrix of h_ϕ at τ. To ensure that G_ϕ is invertible and to facilitate the computation of the Jacobian matrix for high-dimensional distributions, we utilize Normalized Flow models such as NICE (Dinh, Krueger, and Bengio 2014) or Real_NVP (Dinh, Sohl-Dickstein, and Bengio 2016), which have triangular Jacobian matrices, to construct the generator.

However, training Flow-GAN with traditional adversarial training methods often leads to suboptimal performance in term of log-likelihood, resulting in decreased accuracy in density estimation. On the other hand, Flow-GAN trained using MLE may produce samples of lower quality. To strike a balance between these two objectives, a hybrid loss function can be employed for training Flow-GAN:

$$\min_\phi \max_\nu \mathcal{H}(D_\nu, G_\phi) - \lambda\mathbb{E}_{\tau\sim p_{\text{data}}}\left[\log \zeta_\phi(\tau)\right], \quad (3)$$

where $\mathcal{H}(D_\nu, G_\phi)$ can represent any GAN loss function, In our implementation, we use the Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017) loss function. $\lambda \geq 0$ is a hyperparameter that balances adversarial training and MLE. By adjusting λ, it is possible to fine-tune the Flow-GAN according to the specific requirements of the application, effectively managing the trade-off between density estimation accuracy and sample quality.

As depicted on the left side of Figure 1, current methods for sampling OOD data are highly random and lack precise characterization of OOD data, making it difficult to distinguish OOD actions accurately. This imprecision can adversely affect the Q-function values of in-distribution data.

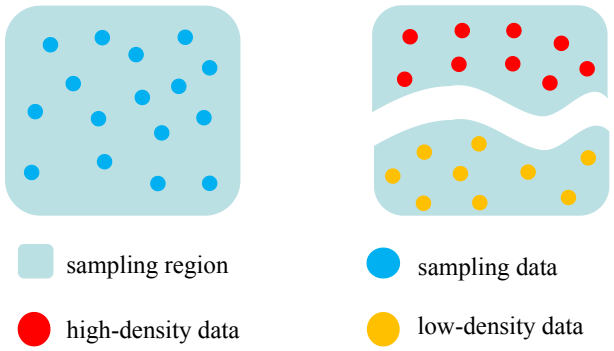


Figure 1: The left panel displays the results of traditional OOD data sampling. The right panel presents the direct density estimation of the sampled data using DT, which classifies the data into high-density and low-density data.

DT uses Equation (2) to directly estimate the density of the sampled data. However, it is difficult to directly determine whether the density exceeds a certain threshold to divide the sampled data. Therefore, we adopt a truncation method to divide the sampled data into two parts. Specifically, we divide the first N highest-density data into high-density data and the remaining into low-density data, as shown on the right side of Figure 1. We can flexibly adjust the size of the high-density data area through N. We only perform OOD regularization on low-density data to avoid harmful effects on the Q-function values of in-distribution data, thereby effectively mitigating extrapolation errors and reducing their impact on uncertainty estimation.

Dynamic Uncertainty Estimation

In offline RL, uncertainty measures can be utilized to perform conservative estimation of the Q-function, thus alleviating extrapolation errors.

We employ the Bootstrapped DQN (Osband et al. 2016) method for uncertainty estimation by integrating multiple Q-networks. Specifically, we use M Q-networks, where $Q_{\theta_m}(s,a)$ represents the Q-function of the m-th network, and $Q_{\theta_{\bar{m}}}(s,a)$ denotes the corresponding target network. The estimation of uncertainty $H(s,a)$ is computed as follows:

$$H(s,a) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(Q_{\theta_m}(s,a) - \bar{Q}(s,a)\right)^2},$$

$$\bar{Q}(s,a) = \frac{1}{M}\sum_{m=1}^{M} Q_{\theta_{\bar{m}}}(s,a).$$

We leverage uncertainty estimation to perform conservative Q-function estimation. For a sample (s,a,r,s') drawn from the offline dataset D, the target value for the Q-function can be expressed as:

$$TQ_\theta^m(s,a) = r(s,a) + \gamma \mathbb{E}_{a'\sim\pi_\psi}\left[\bar{Q}(s',a') - \beta H(s',a')\right], \tag{4}$$

where T denotes the Bellman operator, and β is a learnable parameter.

However, training the Q-function exclusively on offline datasets makes it difficult to effectively regularize OOD actions. To address this issue, we use DT to effectively regularize OOD data. Specifically, we sample states s^{ood} from the offline dataset D and use the current policy π_ψ to sample OOD actions a^{ood}, where $a^{\text{ood}} \sim \pi_\psi(\cdot \mid s^{\text{ood}})$. Then, we use DT to partition the sampled data, removing high-density data to avoid adverse effects on the Q-values of in-distribution data. Since the state transition function $P(\cdot \mid s^{\text{ood}}, a^{\text{ood}})$ and the reward function $r(s^{\text{ood}}, a^{\text{ood}})$ are unknown, we construct an auxiliary target value for the OOD data as follows:

$$\tilde{T}Q_{\theta_m}(s^{\text{ood}}, a^{\text{ood}}) = Q_{\theta_m}(s^{\text{ood}}, a^{\text{ood}}) - \omega H(s^{\text{ood}}, a^{\text{ood}}), \tag{5}$$

where $\tilde{T}$ denotes the Bellman operator, and ω is a learnable parameter. By applying Equation (5), we suppress the Q-function values for OOD actions, thereby effectively regularizing OOD actions and mitigating the impact of extrapolation errors on uncertainty estimation.

By applying Equations (4) and (5), we derive the loss function for each Q_{θ_m} in DURL as follows:

$$\min_{\theta_m} \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(TQ_{\theta_m}(s,a) - Q_{\theta_m}(s,a)\right)^2\right] + \mathbb{E}\left[\left(\tilde{T}Q_{\theta_m}(s^{\text{ood}}, a^{\text{ood}}) - Q_{\theta_m}(s^{\text{ood}}, a^{\text{ood}})\right)^2\right], \tag{6}$$

where $s^{\text{ood}} \sim D, a^{\text{ood}} \sim \pi_\psi$. We obtain the policy π_ψ by solving the following optimization problem:

$$\max_{\psi} \mathbb{E}_{s\sim D, a\sim\pi_\psi}\left[\min_{m=1,\dots,M} Q_{\theta_m}(s,a) - \alpha \log \pi_\psi(\cdot|s)\right]. \tag{7}$$

Due to the inherent randomness in the agent's training process, determining the optimal fixed values for β and ω is challenging. Setting these values too high can lead to an overly pessimistic Q-function, which suppresses the generalization of the Q-function and hinders policy improvement. Conversely, setting them too low makes it difficult to control the overestimation of the Q-function caused by extrapolation errors, significantly impacting policy learning. Therefore, β and ω need to be learnable parameters, allowing for dynamic adjustment during the training process to achieve dynamic uncertainty estimation (DE). We adjust β and ω by minimizing the expected absolute error between the estimated values and the target values. The loss functions for β and ω are defined as follows:

$$\min_{\beta} \ \beta \times \mathbb{E}\left[|TQ_\theta^m(s,a) - Q_\theta^m(s,a)|\right], \tag{8}$$

$$\min_{\omega} \ \omega \times \mathbb{E}\left[\left|\tilde{T}Q_\theta^m(s^{ood}, a^{ood}) - Q_\theta^m(s^{ood}, a^{ood})\right|\right]. \tag{9}$$

During training, we use Equations (8) and (9) to adjust β and ω dynamically. This dynamic uncertainty estimation method regularizes OOD actions while avoiding excessive pessimism and enhancing generalization. The detailed implementation of the algorithm is provided in Appendix C.

		CQL	UWAC	TD3-BC	MCQ	PBRL	CSVE	DURL
Random	HalfCheetah	17.3 ± 1.3	2.3 ± 1.1	11.2 ± 2.1	$\mathbf{28.5 \pm 0.6}$	10.7 ± 3.1	26.7 ± 2.0	19.6 ± 1.8
	Hopper	7.4 ± 0.7	2.5 ± 0.4	8.0 ± 1.4	31.8 ± 0.5	27.4 ± 4.3	27.0 ± 8.5	$\mathbf{33.4 \pm 2.7}$
	Walker2d	5.1 ± 1.4	2.0 ± 0.4	1.6 ± 0.2	$\mathbf{17.0 \pm 3.0}$	8.1 ± 2.8	6.1 ± 0.8	11.3 ± 1.4
Medium	HalfCheetah	44.2 ± 1.4	43.5 ± 0.6	48.3 ± 1.4	64.3 ± 0.2	58.6 ± 1.5	48.6 ± 0.0	$\mathbf{68.8 \pm 1.2}$
	Hopper	52.5 ± 12.5	51.4 ± 0.4	59.3 ± 2.5	74.4 ± 3.8	77.3 ± 21.4	$\mathbf{99.4 \pm 5.3}$	92.4 ± 4.2
	Walker2d	72.8 ± 12.4	73.6 ± 1.8	83.8 ± 1.3	90.3 ± 0.6	89.8 ± 0.4	82.5 ± 1.5	$\mathbf{98.5 \pm 0.8}$
Medium Replay	HalfCheetah	44.5 ± 0.8	35.9 ± 2.8	44.3 ± 0.8	55.4 ± 0.7	43.1 ± 3.3	54.8 ± 0.8	$\mathbf{66.4 \pm 0.4}$
	Hopper	86.8 ± 8.6	26.7 ± 1.8	62.6 ± 17.5	101.4 ± 0.8	100.6 ± 0.8	91.7 ± 0.3	$\mathbf{103.6 \pm 0.7}$
	Walker2d	81.8 ± 1.6	23.6 ± 5.8	87.1 ± 4.8	91.3 ± 4.8	83.7 ± 4.5	78.5 ± 1.8	$\mathbf{98.2 \pm 0.3}$
Medium Expert	HalfCheetah	75.6 ± 4.8	42.7 ± 0.5	90.5 ± 2.8	87.3 ± 1.4	92.3 ± 1.8	93.1 ± 0.3	$\mathbf{104.8 \pm 0.8}$
	Hopper	105.6 ± 10.8	43.9 ± 6.5	96.5 ± 7.8	111.4 ± 0.1	110.6 ± 0.8	95.2 ± 3.8	$\mathbf{113.1 \pm 0.6}$
	Walker2d	106.3 ± 1.8	96.5 ± 9.4	110.1 ± 0.6	114.2 ± 0.6	110.1 ± 0.8	109.0 ± 0.1	$\mathbf{119.5 \pm 0.8}$
Expert	HalfCheetah	96.3 ± 1.2	91.3 ± 0.8	96.4 ± 1.2	96.2 ± 0.4	94.4 ± 1.8	93.8 ± 0.1	$\mathbf{103.4 \pm 2.1}$
	Hopper	96.5 ± 20.4	110.5 ± 0.6	107.5 ± 6.3	111.2 ± 0.4	109.8 ± 0.4	111.2 ± 0.6	$\mathbf{113.6 \pm 1.4}$
	Walker2d	108.5 ± 0.3	108.4 ± 0.6	110.3 ± 0.2	107.2 ± 1.3	108.3 ± 0.2	108.5 ± 0.0	$\mathbf{121.6 \pm 0.7}$
Average		66.7 ± 5.3	50.3 ± 2.2	67.8 ± 3.4	78.8 ± 1.3	75.0 ± 3.2	75.1 ± 1.7	$\mathbf{84.5 \pm 1.3}$

Table 1: Comparison of normalized average scores and standard deviations of all algorithms across four different random seeds on the D4RL MuJoCo "-v2" datasets. The results for CSVE are sourced from the original paper, while results for other baseline algorithms are obtained by running the official code. The highest score for each dataset is highlighted in bold.

		CQL	UWAC	TD3-BC	MCQ	PBRL	CSVE	DURL
Cloned	Pen	39.2 ± 8.4	23.0 ± 6.5	0.0 ± 0.0	49.4 ± 4.3	72.3 ± 9.5	55.2 ± 6.1	$\mathbf{78.3 \pm 10.2}$
	Hammer	2.1 ± 1.1	0.3 ± 0.0	0.0 ± 0.0	$\mathbf{1.3 \pm 0.4}$	0.8 ± 0.4	0.5 ± 0.2	1.1 ± 0.5
	Door	0.4 ± 0.2	0.2 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3.3 ± 4.3	1.3 ± 1.1	$\mathbf{6.5 \pm 2.3}$
	Relocate	-0.1 ± 0.0	0.0 ± 0.0	-0.1 ± 0.0	1.4 ± 0.5	-0.1 ± 0.0	-0.3 ± 0.0	$\mathbf{1.8 \pm 0.4}$
Expert	Pen	107.0 ± 10.4	98.2 ± 8.5	0.3 ± 0.0	80.4 ± 4.2	135.7 ± 3.4	142.9 ± 10.6	$\mathbf{148.5 \pm 3.3}$
	Hammer	86.7 ± 6.2	107.3 ± 25.1	0.0 ± 0.0	50.3 ± 5.2	127.5 ± 0.2	126.5 ± 0.4	$\mathbf{132.5 \pm 5.6}$
	Door	101.5 ± 13.4	104.7 ± 0.4	0.0 ± 0.0	40.5 ± 2.3	96.7 ± 12.2	104.3 ± 0.9	$\mathbf{108.3 \pm 7.2}$
	Relocate	95.0 ± 2.4	105.5 ± 3.2	0.0 ± 0.0	35.6 ± 4.2	84.5 ± 12.2	$\mathbf{103.0 \pm 1.1}$	90.3 ± 1.3
Average		54.0 ± 5.3	54.9 ± 5.5	0.0 ± 0.0	32.4 ± 2.6	65.1 ± 5.3	66.7 ± 2.6	$\mathbf{70.9 \pm 3.9}$

Table 2: Comparison of normalized average scores and standard deviations of all algorithms across four different random seeds on the Adroit tasks. Results for CSVE are taken from the original paper, while results for other baseline algorithms are obtained by running the official code. The highest score for each dataset is highlighted in bold.

Experiment

In this section, we first evaluate our algorithm's performance improvements compared to the latest state-of-the-art (SOTA) algorithms. Next, we applied DT to widely used baselines to validate its effectiveness. Finally, we conduct ablation studies to thoroughly validate the effectiveness of each component of our algorithm. This rigorous validation process provides reassurance about the reliability of our research. Detailed experimental settings can be found in Appendix B.

Performance on MuJoCo Datasets

In the D4RL (Fu et al. 2020) benchmark, we evaluated our algorithm in the Gym-MuJoCo domain and compared it with existing SOTA algorithms. The baseline algorithms we compared against include CQL (Kumar et al. 2020), UWAC (Wu et al. 2021), TD3-BC (Fujimoto and Gu 2021), MCQ (Lyu et al. 2022), PBRL (Bai et al. 2022), and CSVE (Chen et al. 2023). As shown in Table 1, we evaluated the performance of our algorithm and the baseline algorithms on three continuous control tasks: HalfCheetah, Hopper, and Walker2d. Each task has five dataset types: random, medium, medium-replay, medium-expert, and expert, making a total of 15 datasets. Our algorithms were trained for 1 million gradient steps across four different random seeds, with policy evaluations performed every 1000 steps. The final results are the average of the last 10 evaluations. Among the baseline algorithms, MCQ and CSVE demonstrated the best performance. Compared to MCQ, our algorithm outperformed in 13 out of 15 tasks. MCQ showed superior performance only on the Walker2d-random and HalfCheetah-random datasets. When compared to CSVE, our algorithm outperformed in 14 out of 15 tasks, with CSVE showing better performance only on the Hopper-medium dataset. Overall, our algorithm outperformed the baseline algorithms in 12 out of 15 tasks. Notably, our algorithm showed significant improvements over the baseline algorithms on the expert and medium-expert datasets. These results have profound implications for the field of offline RL, as they validate the effectiveness of

		CQL	CQL+DT	PBRL	PBRL+DT	MCQ	MCQ+DT
Random	HalfCheetah	17.3	21.1 (+22.0%)	10.7	15.5 (+44.9%)	28.5	31.2 (+9.5%)
	Hopper	7.4	8.3 (+12.2%)	27.4	29.4 (+7.3%)	31.8	30.5 (-4.1%)
	Walker2d	5.1	10.2 (+100.0%)	8.1	11.4 (+40.7%)	17.0	21.4 (+25.9%)
Medium	HalfCheetah	44.2	51.6 (+16.7%)	58.6	61.3 (+4.6%)	64.3	66.3 (+3.1%)
	Hopper	52.5	61.5 (+17.1%)	77.3	85.2 (+10.2%)	74.4	87.5 (+17.6%)
	Walker2d	72.8	86.3 (+18.5%)	89.8	93.4 (+4.0%)	90.3	94.2 (+4.3%)
Medium Replay	HalfCheetah	44.5	63.8 (+43.3%)	43.1	51.2 (+18.8%)	55.4	59.4 (+7.2%)
	Hopper	86.8	102.4 (+18.0%)	100.6	101.3 (+0.7%)	101.4	102.3 (+0.9%)
	Walker2d	81.8	92.3 (+12.8%)	83.7	89.6 (+7.0%)	87.1	95.3 (+9.4%)
Medium Expert	HalfCheetah	75.6	93.4 (+23.5%)	92.3	98.4 (+6.6%)	87.3	99.7 (+14.2%)
	Hopper	105.6	112.6 (+6.6%)	110.6	112.3 (+1.5%)	111.4	113.7 (+2.1%)
	Walker2d	106.3	111.8 (+5.2%)	110.1	114.5 (+4.0%)	114.2	116.5 (+2.0%)
Expert	HalfCheetah	96.3	95.5 (-0.8%)	94.4	97.1 (+2.9%)	96.2	99.5 (+3.4%)
	Hopper	96.5	113.7 (+17.8%)	109.8	112.2 (+2.2%)	111.2	111.7 (+0.4%)
	Walker2d	108.5	114.3 (+5.3%)	108.3	114.7 (+5.9%)	107.2	115.7 (+7.9%)
Average		66.7	75.9 (+13.8%)	75.0	79.2 (+5.6%)	78.8	83.0 (+5.3%)

Table 3: Comparison of the normalized average scores of CQL, CQL+DT, PBRL, PBRL+DT, MCQ, and MCQ+DT across four different random seeds on the D4RL MuJoCo "-v2" datasets. All algorithms were trained for 1 million gradient steps. (·) represents the percentage improvement relative to the baseline.

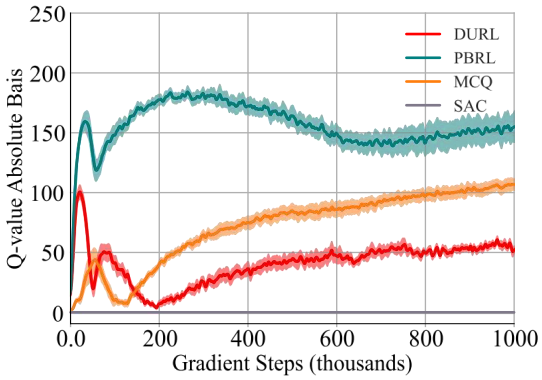


Figure 2: Q-value Absolute Bias of DURL, PBRL, and MCQ, with shaded areas representing standard deviation.

our algorithm and its potential to significantly advance the SOTA. Learning curves can be found in Appendix D.

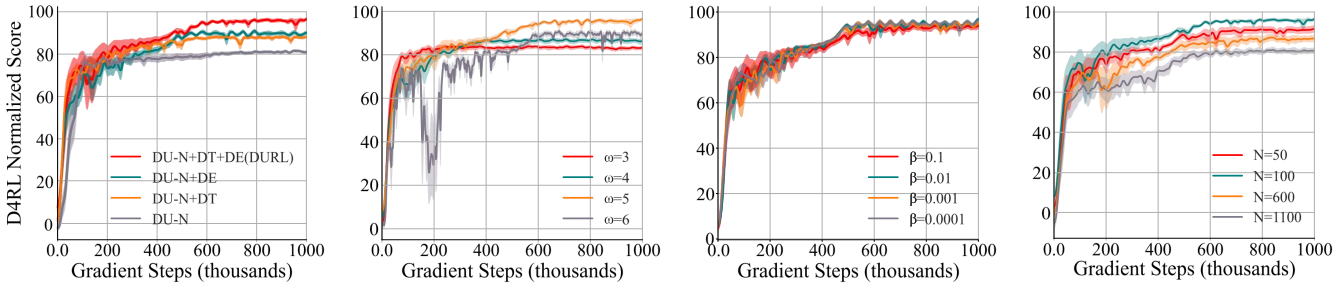
Performance on Adroit and Maze2d Datasets

Adroit tasks. We tested our algorithm on the more challenging Adroit tasks, which consist of four distinct tasks: Pen, Hammer, Door, and Relocate. We conducted experiments using two types of datasets: cloned and expert. Our algorithms underwent training for 1 million gradient steps with four different random seeds, and the policy was evaluated every 1,000 steps. The final results were calculated as the average of the last 10 evaluations and are summarized in Table 2. Out of the eight tasks, our algorithm outperformed the baseline algorithms in six tasks. Notably, our algorithm showed a significant performance improvement over the baseline on the cloned datasets. Moreover, our algorithm generally performed better than the baseline on the expert datasets. Overall, these experimental results validate the effectiveness of our proposed algorithm.

Maze2d. We assessed our algorithm and the baseline algorithms on four datasets: maze2d-umaze, maze2d-umaze-dense, maze2d-medium, and maze2d-medium-dense. The experimental results can be found in Appendix D. We observed that QT (Hu et al. 2024) is the best-performing baseline algorithm, and our method significantly outperforms all baseline algorithms on these datasets.

Q-value Absolute Bias

To clearly demonstrate the effectiveness of our method in avoiding excessive pessimism, we compare Q-function values. Q-function values can reflects whether the Q-function suffers from excessive pessimism. We trained the SAC algorithm for 1 million gradient steps in the Hopper-v2 environment and randomly selected 100 test samples from the online data. Then, we trained PBRL, MCQ, and DURL on the Hopper-medium-replay-v2 dataset, and SAC in the Hopper-v2 environment, with all algorithms being trained for 1 million gradient steps. During training, we recorded Q-function values for the test data every 1,000 steps. We use SAC's estimation on the test data as the true Q-values. We calculated the absolute Q-value bias of PBRL, MCQ, and DURL relative to SAC. The results shown in Figure 2, indicate that DURL's Q-function values are closer to SAC's Q-values. DURL achieved more accurate Q-values compared to the baseline algorithms. The experimental results validate DURL's effectiveness in avoiding excessive pessimism and enhancing Q-function generalization.



(a) Effect of component (b) Effect of ω (c) Effect of β (d) Effect of N

Figure 3: In the ablation study conducted on the Walker2d-medium-v2 dataset, all algorithms were trained for 1 million gradient steps across four different random seeds. Figure 3(a) confirms the effectiveness of the various components of the algorithm. Figure 3(b) analyzes the impact of the parameter ω on the algorithm's performance, Figure 3(c) investigates the effect of the parameter β on performance, and Figure 3(d) explores how the density truncation hyperparameter N influences the algorithm's performance.

Validation of DT Effectiveness

To further validate the effectiveness of the DT, we applied this method to widely used baselines like CQL, PBRL, and MCQ. We tested these algorithms on 15 datasets in the Gym-MuJoCo domain and 8 datasets from Adroit tasks. Each algorithm was trained for 1 million gradient steps with four different random seeds. The final results are the average of the last 10 evaluations. Results in the Gym-MuJoCo domain are shown in Table 3, and the results for Adroit tasks are provided in Appendix D. We observed that applying DT improved the performance of CQL, PBRL, and MCQ. The experimental results suggest that DT effectively regularizes OOD actions and avoids causing harmful effects on in-distribution data.

Ablation Study

To further validate the effectiveness of the components of our proposed algorithm, we conducted ablation studies on the Walker2d-medium-v2 dataset. We refer to the initial version of DURL without the DE and DT components as DU-N. Since DE is not utilized for DU-N, we fixed β in Equation (4) at 0.1 and ω in Equation (5) at 1.0. We then meticulously compared the performance of DU-N with DU-N+DE, DU-N+DT, and DU-N+DT+DE (DURL) variants. The learning curves of these methods are shown in Figure 3(a). The figure illustrates that DU-N+DT+DE (DURL) achieves the best performance. DU-N+DT and DU-N+DE show significant improvements over DU-N, with DU-N+DE outperforming DU-N+DT, indicating that DE contributes more than DT. These results demonstrate the performance comparison among different algorithm variants and the crucial role of DE and DT in our algorithm.

Learnable parameter ω. To evaluate the impact of the initial value of the coefficient ω on algorithm performance, We conducted experiments on the Walker2d-medium-v2 dataset using different initial values of ω. The results are illustrated in Figure 3(b). We observed that the performance was poorest with $\omega = 3$ and best with $\omega = 5$. These findings suggest that both excessively large and excessively small initial values of ω adversely affect the algorithm's performance. An overly large ω can impair the generalization capability of the Q-function, while an excessively small ω fails to effectively regularize the OOD actions.

Learnable parameter β. We tested β on the Walker2d-medium-v2 dataset with values $\{0.1, 0.01, 0.001, 0.0001\}$, as shown in Figure 3(c). The experiments demonstrate that our algorithm exhibits strong robustness with respect to β within the range $[0.0001, 0.1]$.

Density Truncation parameter N. We analyzed the sensitivity of our algorithm to the hyperparameter N in density truncation. We tested various values of N, specifically $\{50, 100, 600, 1100\}$, on the Walker2d-medium-v2 dataset. The results are illustrated in Figure 3(d). The performance is optimal when $N = 100$. In contrast, values of $N = 50$, $N = 600$, and $N = 1100$ all lead to varying degrees of performance degradation compared to $N = 100$. The experimental results indicate that a smaller N fails to filter out in-distribution data, leading to worse performance. Conversely, a larger N does not effectively regularize OOD actions, adversely affecting performance. Therefore, appropriate density truncation is crucial for algorithm performance.

Conclusion

This paper presents a dynamic uncertainty estimation method for model-free offline RL. By integrating dynamic uncertainty estimation and DT, Our method can accurately distinguish OOD actions and dynamically adjust the pessimism of the Q-function, effectively mitigating the overly pessimistic Q-function estimation, thereby improving the generalization of the Q-function. Experimental results demonstrate that our algorithm surpasses existing methods. We validated the effectiveness of each algorithm component and assessed the impact of different hyperparameters on performance. Future work will explore more effective OOD data partitioning mechanisms and dynamic uncertainty estimation methods in model-based offline RL.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2020AAA0106100), the National Natural Science Foundation of China (Nos. 62276160, 62476228), the Natural Science Foundation of Shanxi Province, China (Nos. 202203021211294, 202203021211291), and the Sichuan Science and Technology Program (No. 2024ZYD0180).

References

An, G.; Moon, S.; Kim, J.-H.; and Song, H. O. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *NeurIPS*, volume 34, 7436–7447.

Antoniou, A.; Storkey, A.; and Edwards, H. 2017. Data augmentation generative adversarial networks. *arXiv e-prints*, arXiv–1711.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *ICML*, 214–223. PMLR.

Bai, C.; Wang, L.; Yang, Z.; Deng, Z.; Garg, A.; Liu, P.; and Wang, Z. 2022. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv e-prints*, arXiv–2202.

Cetin, E.; and Celiktutan, O. 2023. Learning pessimism for reinforcement learning. In *AAAI*, volume 37, 6971–6979.

Chen, C.; Modares, H.; Xie, K.; Lewis, F. L.; Wan, Y.; and Xie, S. 2019. Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics. *IEEE Transactions on Automatic Control*, 64(11): 4423–4438.

Chen, H.; Lu, C.; Ying, C.; Su, H.; and Zhu, J. 2022. Offline reinforcement learning via high-Fidelity generative behavior modeling. *arXiv e-prints*, arXiv–2209.

Chen, L.; Yan, J.; Shao, Z.; Wang, L.; Lin, Q.; Rajmohan, S.; Moscibroda, T.; and Zhang, D. 2023. Conservative state value estimation for offline reinforcement learning. In *NeurIPS*, 35064–35083.

Cheng, C.-A.; Xie, T.; Jiang, N.; and Agarwal, A. 2022. Adversarially trained actor critic for offline reinforcement learning. In *ICML*, 3852–3878.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: non-linear independent components estimation. *arXiv e-prints*, arXiv–1410.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2016. Density estimation using real nvp. *arXiv e-prints*, arXiv–1605.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4rl: datasets for deep data-driven reinforcement learning. *arXiv e-prints*, arXiv–2004.

Fujimoto, S.; and Gu, S. S. 2021. A minimalist approach to offline reinforcement learning. *NeurIPS*, 34: 20132–20145.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *ICML*, 2052–2062.

Ghasemipour, S. K. S.; Schuurmans, D.; and Gu, S. S. 2021. Emaq: expected-max q-learning operator for fimple yet effective offline and online rl. In *ICML*, 3682–3691.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPS*, volume 27, 2672–2680.

Grover, A.; Dhar, M.; and Ermon, S. 2018. Flow-gan: combining maximum likelihood and adversarial learning in generative models. In *AAAI*, volume 32.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv e-prints*, arXiv–1812.

Hoel, C.-J.; Wolff, K.; and Laine, L. 2023. Ensemble quantile networks: uncertainty-aware reinforcement learning with applications in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(6): 6030–6041.

Hu, S.; Fan, Z.; Huang, C.; Shen, L.; Zhang, Y.; Wang, Y.; and Tao, D. 2024. Q-value regularized transformer for offline reinforcement learning. *arXiv e-prints*, arXiv–2405.

Janner, M.; Du, Y.; Tenenbaum, J.; and Levine, S. 2022. Planning with diffusion for flexible behavior synthesis. In *ICML*, 9902–9915.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MoreL: model-based offline reinforcement learning. In *NeurIPS*, volume 33, 21810–21823.

Kostrikov, I.; Nair, A.; and Levine, S. 2021. Offline reinforcement learning with implicit q-learning. *arXiv e-prints*, arXiv–2110.

Kumar, A.; Fu, J.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, volume 32, 11784–11794.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, volume 33, 1179–1191.

Lyu, J.; Ma, X.; Li, X.; and Lu, Z. 2022. Mildly conservative a-learning for offline reinforcement learning. In *NeurIPS*, volume 35, 1711–1724.

Nikulin, A.; Kurenkov, V.; Tarasov, D.; and Kolesnikov, S. 2023. Anti-exploration by random network distillation. In *ICML*, 26228–26244.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped dqn. *arXiv e-prints*, arXiv–1602.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv e-prints*, arXiv–1511.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Wei, W.; Zhang, Y.; Liang, J.; Li, L.; and Li, Y. 2022. Controlling underestimation bias in reinforcement learning via quasi-median operation. In *AAAI*, volume 36, 8621–8628.

Wu, F.; Zhang, R.; Yi, Q.; Gao, Y.; Guo, J.; Peng, S.; Lan, S.; Han, H.; Pan, Y.; Yuan, K.; et al. 2024. Ocean-mbrl: offline

conservative exploration for model-based offline reinforcement Learning. In *AAAI*, volume 38, 15897–15905.

Wu, J.; Wu, H.; Qiu, Z.; Wang, J.; and Long, M. 2022. Supported policy optimization for offline reinforcement learning. In *NeurIPS*, volume 35, 31278–31291.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *arXiv e-prints*, arXiv–1911.

Wu, Y.; Zhai, S.; Srivastava, N.; Susskind, J.; Zhang, J.; Salakhutdinov, R.; and Goh, H. 2021. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv e-prints*, arXiv–2105.

Xie, T.; Cheng, C. A.; Jiang, N.; Mineiro, P.; and Agarwal, A. 2021. Bellman-consistent pessimism for offline reinforcement learning. In *NeurIPS*, volume 34, 6683–6694.

Yang, K.; Tao, J.; Lyu, J.; and Li, X. 2024. Exploration and anti-exploration with distributional random network distillation. *arXiv preprint arXiv:2401.09750*.

Yang, S.; Feng, Y.; Zhang, S.; and Zhou, M. 2022a. Regularizing a model-based policy stationary distribution to stabilize offline reinforcement learning. In *ICML*, 24980–25006.

Yang, S.; Wang, Z.; Zheng, H.; Feng, Y.; and Zhou, M. 2022b. A behavior regularized implicit policy for offline reinforcement learning. *arXiv e-prints*, arXiv–2202.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2021. Reinforcement learning in healthcare: A Survey. *ACM Computing Surveys (CSUR)*, 55(1): 1–36.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. Mopo: model-based offline policy optimization. In *NeurIPS*, volume 33, 14129–14142.

Zhang, J.; Lyu, J.; Ma, X.; Yan, J.; Yang, J.; Wan, L.; and Li, X. 2023. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning. *arXiv e-prints*, arXiv–2304.

Zhang, Y.; Li, L.; Wei, W.; Lv, Y.; and Liang, J. 2024a. A unified framework to control estimation error in reinforcement learning. *Neural Networks*, 178: 106483.

Zhang, Y.; Li, L.; Wei, W.; You, X.; and Liang, J. 2024b. Controlling estimation error in reinforcement learning via Reinforced Operation. *Information Sciences*, 675: 120736.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2223–2232.